

Fast Extraction of Adaptive Multiresolution Meshes with Guaranteed Properties from Volumetric Data

Marcel Gavrilu*

Joel Carranza

David E. Breen

Alan H. Barr

Computer Science Department
California Institute of Technology

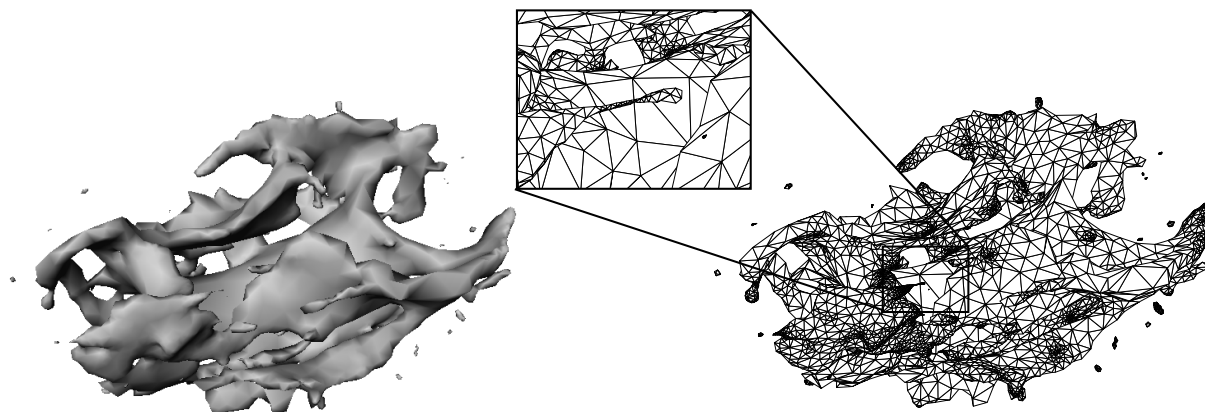


Figure 1: **Examples of meshes created with our algorithm:** The surfaces have very complex structure, high genus, high curvature, and multiple disconnected components. The meshes are the coarsest level generated by our algorithm and have been produced without user intervention. **Left:** smooth shaded view. **Right:** hidden line view. **Inset:** zoomed view highlighting the level of adaptivity generated by the coarse mesh extraction algorithm.

Abstract

We present a new algorithm for extracting adaptive multiresolution triangle meshes from volume datasets. The algorithm guarantees that the topological genus of the generated mesh is the same as the genus of the surface embedded in the volume dataset at all levels of detail. In addition to this “hard constraint” on the genus of the mesh, the user can choose to specify some number of soft geometric constraints, such as triangle aspect ratio, minimum or maximum total number of vertices, minimum and/or maximum triangle edge lengths, maximum magnitude of various error metrics per triangle or vertex, including maximum curvature (area) error, maximum distance to the surface, and others. The mesh extraction process is fully automatic and does not require manual adjusting of parameters to produce the desired results as long as the user does not specify incompatible constraints. The algorithm robustly handles special topological cases, such as trimmed surfaces (intersections of the surface with the volume boundary), and manifolds with multiple disconnected components (several closed surfaces embedded in the same volume dataset). The meshes may self-intersect at coarse resolutions. However, the self-intersections are corrected automatically as the resolution of the meshes increase. We show several

*mg@cs.caltech.edu

examples of meshes extracted from complex volume datasets.

1 Introduction

Volumetric datasets occur in many important applications, such as medical imaging, fluid dynamics simulation, 3D laser data scanning, and others. Consequently, methods for visualizing volume datasets are an important topic in computer graphics and have been the subject of extensive research. Two main approaches for visualizing volume data have been proposed, volume rendering and iso-surface extraction. Among the two, iso-surface extraction in the form of triangle meshes is the most widely used.

These extracted meshes should ideally possess a number of important properties. The meshes should be adaptive, adjusting triangle size to local curvature in order to adequately represent the surface shape with a minimum number of triangles. Keeping triangle count to a minimum is critical when attempting to interactively view or edit large models. A multiresolution representation of the surface is also important during interactive display or transmission of large models. The extracted surface should be represented by a number of meshes with varying triangle counts. All the vertices present in a coarse mesh should also be present in the next higher-resolution mesh. New vertices are added to increase resolution and connectivity is recomputed locally. In many medical and scientific applications it is essential that the extracted meshes represent the object of interest as accurately as possible. Therefore the mesh should capture the correct topology of the sampled object at all resolutions and converge to the surface as they are refined. When objects are scanned they may consist of multiple components or may only be partially scanned, i.e. only parts of the object lay inside the volume. The iso-surface extraction algorithm should therefore be able to extract both closed and open (*trimmed*) surfaces, possibly consisting of multiple disconnected components. Finally, the aspect ratios (ratio of shortest edge to the longest edge) of the individ-

	T	F	OS	AR	AD	MR	SD	CI
LORENSEN & CLINE (MC) [14]	✓	✓	✓					○
MÜLLER & STARK [18]	○		✓		✓			○
WESTERMANN ET AL. [23]	○	✓	✓		✓			○
WYVILL ET AL. [26]	○	✓	✓					○
BLOOMENTHAL [1]	✓	✓	✓					✓
HARTMANN [7]	○	✓	✓	✓				○
HENDERSON [8]	○	✓	✓	✓				○
KARKANIS & STEWART [10]	○	✓	✓	○	✓			○
MILLER ET AL. [17]				○			✓	○
LACHAUD & MONTANVERT [12]	○			○		✓		○
VELHO ET AL. [22]	○				✓	✓		○
WOOD ET AL. [25]	✓			○	✓	✓	✓	○
MC + SIMPLIFICATION [19, 9, 6]	○		✓		✓	✓		○
MC + REMESHING [20, 5, 11, 13]	○		○	○	✓	✓	✓	○
OUR ALGORITHM	✓	✓	✓	✓	✓	✓		✓

Table 1: **Comparison of iso-surface extraction algorithms:** Algorithm/mesh properties: T - captures correct topology (genus), F - fast (seconds to minutes, comparable to marching cubes), OS - capable of extracting an open surface (shell), AR - triangles with good (close to 1) aspect ratios, AD - creates an adaptive mesh, SD - creates a mesh with subdivision connectivity, and CI - mesh converges to correct iso-surface (no self intersections or holes). Empty box - property not provided. Small circle (○) - property provided under certain circumstances. Check (✓) - property guaranteed.

ual triangles in the mesh should be bounded by a reasonable lower limit. Long and thin triangles can cause problems in downstream applications (e.g. finite element analysis), and therefore should not be present in the mesh.

Many different algorithms for extracting triangle meshes from volumetric data have been proposed. None can guarantee to produce adaptive multiresolution meshes with correct topology, bounded triangle aspect ratios, and subdivision connectivity in the presence of boundaries or multiple disconnected components. Many of the algorithms can provide a subset of these properties, but may require significant user intervention in order to converge to the correct iso-surface, restricting their applicability to processing datasets with limited complexity.

We present a new algorithm for directly extracting adaptive multiresolution meshes from volumetric datasets. The algorithm guarantees the following properties:

- generates meshes with the correct topological genus at all resolutions,
- generates meshes that are adaptive at all resolutions,
- extracts trimmed surfaces (boundaries) and multiple disconnected components,
- generates meshes with minimum triangle aspect ratio exceeding 0.3 as mesh resolution increases, and
- the meshes generated automatically converge (error converges to zero) to the iso-surface.

The paper proceeds as follows. Section 2 summarizes previous work in the area of mesh extraction. It compares the properties of our algorithm with other related algorithms. Section 3 includes a high-level summary followed by a detailed discussion of our algorithm. Section 4 presents the results of applying our algorithm on a number of volume datasets. The paper closes with thoughts on future work in Section 5.

2 Previous Work

Numerous algorithms have been developed to extract polygonal meshes from volumetric datasets. Another closely related field focuses on converting implicit models into polygonal meshes. All of these methods may be placed in three broad categories, exhaustive search methods, continuation methods and deformable model methods. Hybrid methods that use one of the previous types of extraction followed by mesh simplification or optimization also exist. Many of these methods are presented in Table 1 with a checklist of associated properties.

Exhaustive Search Methods. The Marching Cubes (MC) algorithm [14] has been the most widely used algorithm for extracting

iso-surfaces from volume data. It examines every cube of data (defined by eight neighboring voxels) in the volume and fits a small number of triangles to the iso-surface within the cube. While the Marching Cubes algorithm is straightforward and easy to implement, it has several known problems. Mainly it creates an enormous number of triangles, many of which have small, troublesome aspect ratios. Since the algorithm acts locally it cannot resolve ambiguities in the data, producing holes in the final mesh under certain circumstances. Several ways of dealing with these ambiguous cases and producing meshes without holes are discussed in [1]. Müller and Stark [18], and Westermann et al. [23] addressed the problem of the large number of triangles in MC meshes by developing versions of the Marching Cubes algorithm that generate adaptive meshes.

Continuation Methods. Continuation methods start at a point on the iso-surface and grow the mesh from that point out to the rest of the iso-surface. Wyvill et al. [26] presented a continuation method for implicit models, which can easily be applied to volumetric data. Bloomenthal [1] presents another variation on this approach that incorporates a method for coping with local iso-surface ambiguities; thus producing a mesh that faithfully captures the topology (genus) of the implicit model. Hartmann [7], Henderson [8], and Karkanis and Stewart [10] present continuation methods that conceptually move a disk along the iso-surface. The neighborhood information derived from the disk produces the samples and vertex connectivity needed to generate triangles with good aspect ratios. Karkanis and Stewart allow their disk to change its radius based on surface curvature. While this allows for an adaptive surface, it weakens the aspect ratio properties of the mesh.

Deformable Model Methods. Miller et al. [17] present the first deformable model for extracting 3D meshes from volume data. Their method was limited to meshes with spherical topology. Lachaud and Montanvert [12] extend this work with methods that change the topology of the mesh and produce multiresolution results. Velho et al. [22] present a general, two-step approach to extracting meshes from volume data. A base mesh that captures the topology of the object of interest is first created, followed by the subdivision and deformation of the base mesh in order to produce the final adaptive, multiresolution mesh. Wood et al. [25] improve upon this approach with a method that guarantees that the final mesh has the correct topology and has subdivision connectivity. Others [21, 2, 16] have developed deformable models for extracting surfaces from volume data, but have not presented information about the properties of their resulting meshes. These approaches have therefore not been included in Table 1.

Hybrid Methods. Other approaches to generating adaptive, multiresolution meshes from volume data can involve first extracting a mesh with the Marching Cubes algorithm, followed by a mesh simplification [19, 9, 6] or a remeshing [20, 5, 11, 13] step. While these approaches may produce the desired result, we feel that they are not optimal. We contend that it is better to extract a mesh with the desired properties directly from the volume, rather than produce a poor-quality mesh that is “fixed” by post-processing. Direct extraction of high-quality meshes from volumes also requires less computation time and memory resources.

The previous methods have been collected and compared in Table 1. The various approaches are listed on the left side. The properties of the algorithms and resulting meshes are listed across the top, T - captures correct topology(genus), F - fast (execution times of seconds to minutes, comparable to those of marching cubes), OS - capable of extracting an open/trimmed surface (shell), AR - triangles with good (close to 1) aspect ratios, AD - creates an adaptive mesh, SD - creates a mesh with subdivision connectivity, and CI - mesh converges to correct iso-surface and has no self-intersections or holes. Each property is evaluated for each approach. If the associated box is empty, the approach does not provide the property. If it contains a small circle (○) it is possible for the method to pro-

vide the property under certain circumstances or if parameters are properly “tweaked.” For example deformable model type methods using distance functions to the iso-surface cannot “fix” (deform) self-intersecting meshes automatically, and require a seed mesh that is self-intersection free. A check (\checkmark) signifies that the approach is guaranteed to provide the associated property. It can be seen that our iso-surface extraction method is guaranteed to provide almost all of the properties of the previous methods. Our approach will capture the correct topology, is capable of extracting open surfaces, generates triangles with good aspect ratio (between 1/3 and 1), the resulting mesh is adaptive and multiresolution, and our mesh will converge to the correct iso-surface. The meshes may initially have self-intersections at coarser resolutions. However, the self-intersections disappear automatically as resolution of the meshes increases.

3 The Algorithm

3.1 Overview

There are many possible ways a surface can intersect the faces of a voxel. A non-ambiguous set of cases describing such intersections has been described by Bloomenthal, and extends that introduced by Lorensen and Cline [14]. Iso-surfaces that can be extracted from volume datasets using this set of heuristics are manifolds embedded in three dimensional Euclidean space. They have no self-intersections or holes because none of the allowed cases for surface to voxel intersections have self-intersections, and the extensions described by Bloomenthal were exactly aimed at eliminating holes in the surface. Such manifolds exhibit many useful properties. In particular, they can be locally parameterized with simple maps. There exists a finite collection of such maps that together create what is called an atlas, or coordinate chart of the manifold. For a more mathematical treatment of manifolds and atlases see [4].

Our algorithm creates triangle meshes from special atlases of the manifold. These special atlases are comprised of maps that are isomorphic to planar disks. The first stage of our algorithm, the adaptive coarse mesh extraction stage, finds such an atlas that has a very small number of disks and then converts it into a mesh. Subsequently, during the adaptive refinement stage, this atlas is modified by halving the radii of the disks covering areas with significant surface detail, and by adding more disks to complete the atlas. This procedure is recursive and continues until user specified constraints are met, conflicting constraints are detected, or the maximum resolution is attained.

```

EXTRACT( VolumeData VD, Metric E, maxError  $\epsilon$  )

// initialization
parse the volume dataset VD and produce the vertex connectivity graph G

// STAGE 1
// extract coarse adaptive mesh with correct topology and nice triangles
DC = EXTRACTCOARSEMESH(G)
// STAGE 2
// refine coarse mesh to desired accuracy
DC = REFINE MESH(E,  $\epsilon$ , G, DC)

build surface Voronoi Neighborhoods of DC
take the dual of the Voronoi neighborhood and build the mesh M
return M

```

Figure 2: Pseudocode of the mesh extraction algorithm.

Stage one, the **adaptive coarse mesh extraction stage**, computes a coarse adaptive mesh with the correct global topology. The vertices of the mesh are chosen from a larger set of candidate vertices according to a number of local topological constraints that ensure that the global topology of the surface is preserved. The associ-

ated surface Voronoi diagram uniquely determines the connectivity of the mesh. In addition to requiring that the global topology be preserved the user may choose to specify other constraints, such as maximum curvature per triangle edge, maximum edge length, minimum number of triangles, etc. The adaptive coarse mesh extraction stage is described in detail in Section 3.3.

During stage two, the **adaptive refinement stage**, vertices of the previously generated coarse mesh are sorted according to one of several error metrics. The vertices whose error is greater than a user controlled threshold are subdivided. The meshes are locally retriangulated by updating the affected surface Voronoi cells and re-connecting the corresponding edges. The aspect ratios of the triangles are controlled by the same edge length constraints used during stage one. The details of the adaptive refinement stage are discussed in Section 3.4.

3.2 Preliminary Definitions and Concepts

The input volume data to our algorithm consists of scalar values at the vertices of a regular 3-dimensional grid. The volume data can be supersampled as a pre-process to increase the resolution of the final mesh. Tri-linear or tri-cubic interpolation work well for this purpose.

Surfel and Candidate Vertices. A *surfel* is a surface patch formed by the intersection of the implicit iso-surface with a voxel as defined in [25]. The intersection of a surfel with a voxel edge is called a *node*. A *candidate vertex* is associated with each surfel. The coordinates of a c-vertex are computed as the average of the coordinates of the nodes of the associated surfel. We define a neighborhood relation between c-vertices: two c-vertices are neighbors if their associated surfels share at least one node.

Candidate Vertex Connectivity Graph. The neighborhood relation between c-vertices is used to build a weighted *candidate vertex (CV) connectivity graph* G . The edges in the graph correspond to pairs of neighboring candidate vertices. The weight of an edge is equal to the Euclidean distance between the c-vertices it connects. The graph completely encodes the topology of the iso-surface. It is similar to the graphs used in [25], except the edges are weighted with real distance information, instead of setting all the edge lengths to 1.

CV Connectivity Graph Geodesics. If v and w are two candidate vertices from the graph G , the $[v, w]$ -*geodesic* is defined as the shortest path in G between v and w . If x and y are c-vertices on some geodesic, then the path between x and y on that geodesic is an $[x, y]$ -geodesic. Note that given three c-vertices u , v , and w , the union of the $[u, v]$ and the $[v, w]$ geodesics is not necessarily a $[u, w]$ -geodesic.

Disk Neighborhoods. Given a candidate vertex $v \in G$ we define the v_α -*disk neighborhood* as the set of all c-vertices $w \in G$ such that the $[v, w]$ -geodesics have length less than α . α is called the radius of the disk neighborhood D_v^α . A v_α -disk neighborhood D_v^α is called *simple* if for all $w \in D_v^\alpha$ there exists exactly one $[v, w]$ -geodesic. It is easy to show that simple neighborhoods have exactly one boundary and are homeomorphic to a planar disk. Two disk neighborhoods D_v^α and D_w^γ are called *independent* if $v \notin D_w^\gamma$ and $w \notin D_v^\alpha$. Two disk neighborhoods are called *interacting* if they are not independent.

Disk Coverings. A *disk covering*, $\mathcal{C}(G)$, of a vertex connectivity graph G is a set of pairs $[v, \alpha]$ of candidate vertices $v \in G$ and real numbers such that for any c-vertex w of G there exists $[v, \alpha] \in \mathcal{C}(G)$ such that $w \in D_v^\alpha$. A disk covering $\mathcal{S}(G)$ is called *simple* if for any $[v, \alpha] \in \mathcal{S}(G)$ the disk neighborhood D_v^α is simple. A disk covering is called *independent* if all its disk neighborhoods are pairwise independent.

Voronoi Neighborhoods. If $\mathcal{C}(G)$ is a covering of G and $[v, \alpha] \in \mathcal{C}(G)$, we define the $\mathcal{C}(G)$ -*Voronoi neighborhood* of v as the set of all c-vertices $w \in G$ such that for any $[u, \gamma] \in \mathcal{C}(G)$ the length of

```

EXTRACTCOARSEMESH(Vertex Connectivity Graph  $G$ )

// initialization
const  $R_g$  // for the max allowed radius of disk neighborhoods
// initialize empty storage objects
set up the bins  $B_i = \emptyset$  // bin for candidate vertices where disks of radius  $\frac{R_g}{2^i}$  may be added
set up the list of forbidden c-vertices  $F = \emptyset$ 
set up the disk covering data structure  $DC = \emptyset$ 
place all vertices of  $G$  in the bin  $B_0$ 

// find a simple independent covering of  $G$ 
while  $\exists B_i$  not empty
    remove a c-vertex  $v$  from lowest numbered non-empty bin  $B_i$ 
     $R_v = \frac{R_g}{2^i}$ 
    // find the largest radius simple disk neighborhood at  $v$ 
    while FINDTOPOLOGICALEVENT( $D_v^{R_v}$ ) // See Figure 4.
         $R_v = \frac{R_v}{2}$  // halve the radius
    endwhile
    // classify the points in  $D_v^{3R_v}$ 
    CLASSIFY( $R_v, D_v^{3R_v}, F, B_i, B_{i-1}$ ) // See Figure 5.
    // check intersecting neighbors for radius consistency
    ENFORCERADIUSCONSTRAINTS( $R_v, D_v^{R_v}, DC$ ) // See Figure 7.
    add  $D_v^{R_v}$  to the disk covering  $DC$ 
endwhile
return  $DC$ 

```

Figure 3: **Stage one of the mesh extraction algorithm:** extracting a coarse mesh with guaranteed global topology.

the $[v, w]$ -geodesic is less than the length of the $[u, w]$ -geodesic. The $\mathcal{C}(G)$ -Voronoi neighborhood of a c-vertex v is denoted by $V_v^{\mathcal{C}(G)}$. The set of all the Voronoi neighborhoods associated with the c-vertices of $\mathcal{C}(G)$ is called the *Voronoi covering* associated with the disk covering $\mathcal{C}(G)$ and is denoted by $V^{\mathcal{C}(G)}$. A Voronoi covering is called simple and/or independent if it is associated with a simple and/or independent disk covering.

Two $\mathcal{C}(G)$ -Voronoi neighborhoods $V_v^{\mathcal{C}(G)}$ and $V_w^{\mathcal{C}(G)}$ are called *adjacent* if there exist c-vertices $x \in V_v^{\mathcal{C}(G)}$ and $y \in V_w^{\mathcal{C}(G)}$ such that x and y are neighbors in G . The set of c-vertices $x \in V_v^{\mathcal{C}(G)}$ with the property that there exists $y \in V_w^{\mathcal{C}(G)}$ such that x and y are neighbors in G is called the *w-boundary* of the $V_v^{\mathcal{C}(G)}$ Voronoi neighborhood. There exists a matching *v-boundary* of the $V_w^{\mathcal{C}(G)}$ Voronoi neighborhood. The two boundaries are always disjoint.

A c-vertex x in $V_v^{\mathcal{C}(G)}$ is called a *Voronoi vertex* if it belongs to at least two separate boundaries of $V_v^{\mathcal{C}(G)}$. A Voronoi neighborhood is *proper* if all its Voronoi vertices belong to exactly two boundaries.

3.3 Stage 1: Coarse Adaptive Mesh Extraction

The coarse mesh extraction algorithm takes in the vertex connectivity graph G and returns a simple disk covering DC of it. The covering is converted into a mesh by first computing its associated Voronoi covering and then building the dual, surface Delaunay triangle mesh.

```

FINDTOPOLOGICALEVENT( Disk Neighborhood  $D_v^{R_v}$  )

// initialization
 $R = 0$ 
// initialize empty storage objects
setup the list of visited c-vertices  $V = \emptyset$ 
add  $v$  to the list of propagating boundary  $P$ 

// propagate out to  $R_v$  and check for boundary self intersections
while  $P$  not empty
    remove a c-vertex  $w$  from front of  $P$ 
    if the neighbors of  $w$  in  $V$  are connected and  $\exists u \in P$  not a neighbor of  $w$ 
        // no boundary self-intersection or boundary vanishing at  $w$ 
        append all neighbors of  $w$  from  $D_v^{R_v} \setminus V$  to  $P$  and  $V$ 
    else
        // boundary self-intersects or vanishes at  $w$ 
        return TRUE
    endif
endwhile
return FALSE

```

Figure 4: **Function for detecting topological events:** boundary splits (self-intersections) and boundary vanish events. Called by the EXTRACTCOARSEMESH algorithm, Figure 3 and the REFINEMESH algorithm, Figure 8.

We achieve the desired mesh properties by enforcing a set of

local topological constraints on the mesh vertices. The first topological condition is that the set of mesh vertices induces a simple independent disk covering $\mathcal{S}_0(G)$ on the vertex connectivity graph. The second topological constraint is that the radii of any two intersecting disk neighborhoods in $\mathcal{S}_0(G)$ do not differ by more than a factor of two.

The simple disk covering is constructed incrementally as presented in Figure 3. The values of the disk radii are restricted to be $R_g/2^i$, where R_g is the maximum radius allowed and i is a non-negative integer. New disks are added to the covering one by one, with the maximum size that preserves the radius constraints between intersecting disks. If a topological event is detected while a new disk is added to DC the size of its radius is halved until the topological event disappears.

```

CLASSIFY( Radius  $R_v$ , Neighborhood  $D_v^{R_v}$ , ForbiddenList  $F$ , Bin  $B_i$ , Bin  $B_{i-1}$  )

// classify points in  $D_v^{R_v}$  according to their distance from  $v$ 
for all points  $w$  in  $D_v^{R_v}$ 
    if  $[v, w]$ -graph geodesic is shorter than  $R_v$ 
        add  $w$  to forbidden list  $F$ 
    else if  $[v, w]$ -graph geodesic is shorter than  $2R_v$  and  $w \notin B_i, j > i$ 
        add  $w$  to bin  $B_i$ 
    else if  $[v, w]$ -graph geodesic is shorter than  $3R_v$  and  $w \notin B_j, j > i-1$ 
        add  $w$  to bin  $B_{i-1}$ 
    endif
endfor

```

Figure 5: **Function for classifying points in the neighborhood of a selected c-vertex.** Called by the EXTRACTCOARSEMESH algorithm, Figure 3 and the REFINEMESH algorithm, Figure 8.

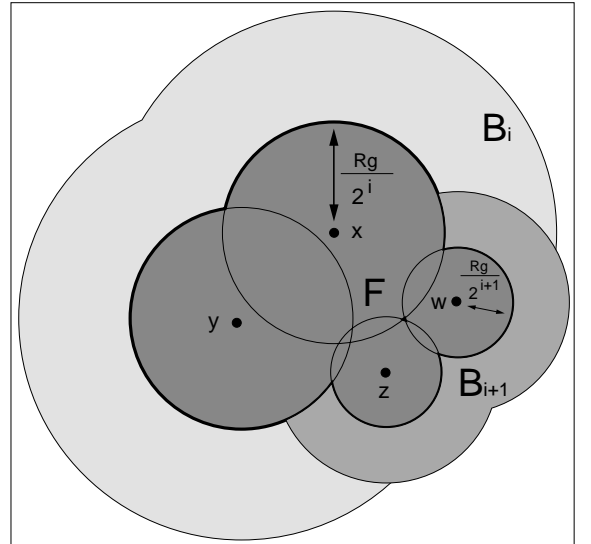


Figure 6: **Classifying vertices with respect to their distance from disk neighborhood centers:** F is the forbidden area (dark gray); a c-vertex within the radius of disk neighborhoods of selected c-vertices is classified as forbidden. B_i, B_{i+1} : If a c-vertex is within twice the radius from the center of the disk neighborhood of a selected c-vertex $d_w^{R_g/2^{i+1}}$ it is classified in the corresponding bin B_{i+1} . If the c-vertex is within twice the radius from the centers of two different disk neighborhoods it will be classified according to the rules set by the neighborhood with the smaller radius.

The algorithm for detecting topological events inside a disk is shown in Figure 4. Starting with the center of the disk, a wavefront is propagated outwards in the disk neighborhood. A topological event is detected at a c-vertex v if the propagating boundary self-intersects or vanishes at v .

We maintain several auxiliary data structures to help us quickly create new disks that are consistent with the constraints listed above. C-vertices of G can be classified as either **forbidden** or be sorted into **bins** B_i . C-vertices are classified as forbidden if they are contained in the interiors of some disk neighborhood from the

partial covering DC . No new disk neighborhoods with centers in the forbidden areas can be added to the partial covering DC without violating the above constraints. The bins B_i contain c-vertices where a disk of radius $R_g/2^i$ can be added without violating the ratio constraints with respect to the partial covering DC , see Figure 6. The pseudocode of the algorithm for the classification of c-vertices into bins is presented in Figure 5.

```

ENFORCERADIUSCONSTRAINTS( Neighborhood  $D_v^{R_v}$ , DiskCovering  $DC$  )

// find all disk neighborhoods of c-vertices in  $DC$  that intersect  $D_v^{R_v}$ 
while  $\exists D_w^{R_w}, w \in DC, D_v^{R_v} \cap D_w^{R_w} \neq \emptyset$ 
  if  $R_v > 2R_w$ 
     $R_v = \frac{R_v}{2}$  // halve the larger radius
    CLASSIFY( $R_v, D_v^{2R_v}, F, B_i, B_{i-1}$ ) // See Figure 5.
    // smaller radius, restart checking constraints for v
    ENFORCERADIUSCONSTRAINTS( $D_v^{R_v}, DC$ )
    break
  else  $2R_v < R_w$ 
     $R_w = \frac{R_w}{2}$  // halve the larger radius
    CLASSIFY( $R_w, D_w^{2R_w}, F, B_i, B_{i-1}$ ) // See Figure 5.
    // smaller radius, need to recheck constraints for w
    ENFORCERADIUSCONSTRAINTS( $D_w^{R_w}, DC$ )
    // finish checking remaining constraints for v
    ENFORCERADIUSCONSTRAINTS( $D_v^{R_v}, DC$ )
    break
  endif
endwhile

```

Figure 7: Function for enforcing that intersecting disk neighborhoods have radii different by at most a factor of two. Called by the EXTRACTCOARSEMESH algorithm, Figure 3 and the REFINEMESH algorithm, Figure 8.

Disks sizes are halved when violations of the ratio constraint between the radii of intersecting neighbors or topological events are detected. Halving one radius can create further violations of the ratio constraint with adjacent neighbors, causing them to halve their radius as well. This chain reaction will eventually stop because radii will never become smaller than the smallest of the radii of the original pair of disk neighborhoods that started it. The algorithm for enforcing radius constraints is shown in Figure 7.

3.4 Stage 2: Adaptive Refinement

For the refinement stage the user specifies the maximum mesh error desired, ϵ . We have tested several different error metrics, described in Section 3.5. The refinement algorithm finds the disk neighborhood $D_v^{R_v}$ with the largest error. If it is larger than ϵ then $D_v^{R_v}$ is halved and reclassified. New disk neighborhoods are added to the covering the same way they were added during the coarse mesh creation stage and the error metric is recomputed on the newly added or resized neighborhoods. The algorithm loops until all the disk neighborhoods in DC have error less than ϵ or they cannot be subdivided any further due to the inherent resolution limitation of the volume dataset. In order to preserve the aspect ratio guarantees, the smallest disk neighborhoods $D_v^{R_v}$ allowed must at least contain all the neighbors of v in the vertex connectivity graph G .

The refinement stage can be repeated several times, with different maximum error requirements, therefore producing a set of multiresolution meshes. The error metrics can be switched on the fly to optimize for different applications.

3.5 Error Metrics for Refinement

We have experimented with a variety of error metrics. Each one has its own advantages and disadvantages and can be used to generate meshes optimized for certain applications.

Maximum Distance From the Surface. This is perhaps the simplest metric. Given a disk neighborhood $D_v^{R_v}$ the metric is computed as the maximum distance from all c-vertices in $D_v^{R_v}$ to their respective closest triangle from the mesh that has a vertex at v or one of its neighbors in DC . The maximum distance metric is well

```

REFINEMESH( Metric  $E$ , MaxError  $\epsilon$ , ConnectivityGraph  $G$ , DiskCovering  $DC$  )

// subdivide all disk neighborhoods that have error more than  $\epsilon$ 
while  $\exists v \in DC, E(D_v^{R_v}) > \epsilon$ 
  // subdivide the neighborhood: halve the radius
   $R_v = \frac{R_v}{2}$ 
  // classify the c-vertices in  $D_v^{2R_v}$ 
  CLASSIFY( $R_v, D_v^{2R_v}, F, B_i, B_{i-1}$ ) // See Figure 5.
  // check intersecting neighbors for radius consistency
  ENFORCERADIUSCONSTRAINTS( $R_v, D_v^{R_v}, DC$ ) // See Figure 7.
  add [ $v, R_v$ ] to  $DC$ 

  // add new vertices in the space created
  while  $B_i$  not empty
    remove a vertex  $w$  from lowest numbered non-empty bin  $B_i$ 
     $R_w = \frac{R_g}{2^{i+1}}$ 
    // no need for more topology checking
    // classify the points in  $D_w^{2R_w}$ 
    CLASSIFY( $R_w, D_w^{2R_w}, F, B_i, B_{i-1}$ ) // See Figure 5.
    // check intersecting neighbors for radius consistency
    ENFORCERADIUSCONSTRAINTS( $R_w, D_w^{R_w}, DC$ ) // See Figure 7.
    add [ $w, R_w$ ] to  $DC$ 
  endwhile
endwhile
return  $DC$ 

```

Figure 8: Stage two of the mesh extraction: adaptive refinement.

suited for refining mesh outlines. It tends to sample more uniformly than curvature based metrics.

Average Distance From the Surface. This metric is similar to the maximum distance metric except that instead of taking the maximum over all the c-vertices in a disk the average of all the distances is computed. Its behavior is similar to that of the maximum distance metric. It tends to subdivide more in areas with large features and will smooth out some of the smaller features of the surfaces.

Area Curvature 1. We estimate surface area and normal vectors for each c-vertex of G during the setup stage of the algorithm. This allows us to compute the total mean surface curvature K (variation of the normal) over a disk neighborhood $D_v^{R_v}$ by taking the ratio between the sums of the areas of the surfels that are covered by $D_v^{R_v}$ and the sum of the same areas projected onto their respective closest triangles in the current mesh. Below, N_v is the normal at v , and N_t is the normal of the triangle closest to v :

$$K = \frac{\sum_{v \in D_v^{R_v}} A_v}{\sum_{v \in D_v^{R_v}} A_v \frac{N_v \cdot N_t}{\|N_v\| \cdot \|N_t\|}}$$

This metric is best for refining around areas of high curvature and, in combination with the distance metric, produces meshes optimized for flat shading.

Area Curvature 2. We present another approach for approximating total mean curvature over a disk neighborhood. Instead of using the same normal across the whole surface of a triangle in our mesh, the previous metric can be modified to use interpolated normals at the point (on that triangle) closest to the surfel being projected. This procedure is justified by the way meshes are rendered. Flat shading uses one normal for the whole triangle, while smooth shading interpolates the normal across the surface of the triangle.

$$K = \frac{\sum_{v \in D_v^{R_v}} A_v}{\sum_{v \in D_v^{R_v}} A_v \frac{N_v \cdot \text{Interp}(N_t)}{\|N_v\| \cdot \|\text{Interp}(N_t)\|}}$$

This metric will still refine areas of high curvature, but will produce fewer triangles than the first curvature metric. The meshes generated look good when smooth-shaded but may have jagged outlines. The outlines can be improved by combining it with the maximum distance metric.

Directional Curvature Metrics. Directional curvature can be estimated by taking the ratios of the geodesic distance to the Euclidean distance between the center of a disk neighborhood and its boundary c-vertices. A better method for computing line curvatures is to sum the angles between the normals of consecutive c-vertices along a geodesic.

3.6 Guaranteeing the Topology

Consider three dimensional implicit iso-surfaces that are 2-manifolds embedded in \mathbf{R}^3 . From its definition it follows that a simple disk covering is an atlas (coordinate chart) of such iso-surface manifolds, and therefore must have the correct topology. In finding this covering we do not explicitly compute the genus of the surface. The genus can be obtained by taking the Euler characteristic of the generated mesh, see [4].

Many algorithms for mesh extraction from volumetric data determine the genus of the iso-surfaces first and then construct a mesh with the same genus which is later deformed and refined to fit the surface. As a result these algorithms have trouble dealing with surfaces with boundaries and often have no way of automatically adjusting the resolution of the meshes to represent fine features of the iso-surfaces. The main body of the dendrite, see Section 4, is topologically equivalent to a sphere, yet it is hardly well represented by a tetrahedron. To make things worse, many of the solvers used in the refinement stages of these algorithms do not allow edges of triangles to move through the iso-surface, see Figure 10, and require that the starting coarse meshes be similar in configuration to the iso-surfaces.

Our method is more robust and does not have the above problems. It generates meshes with the correct global topology while adaptively adjusting its resolution to capture the relevant details present in the data with the minimum amount of triangles, see Section 3.8.

3.7 Constraining the Triangle Aspect Ratios

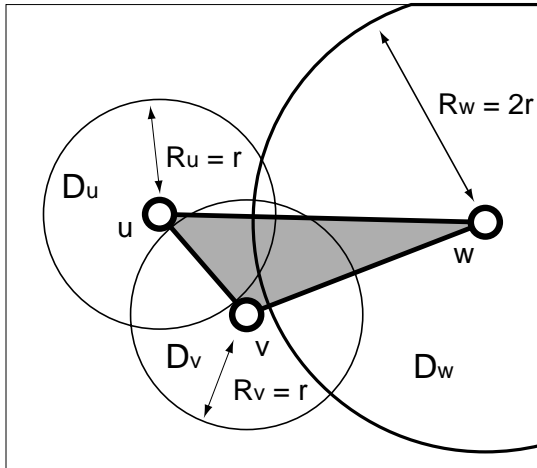


Figure 9: **Constraining triangle aspect ratios:** If three disk neighborhoods $D_u^{R_u}$, $D_v^{R_v}$, and $D_w^{R_w}$ are intersecting but do not contain each other's centers, and the ratio between their radii is between 1/2 and 2 the aspect ratio of the shaded triangle with vertices at the disk centers is better than 1/3.

The constraints on the disk covering also ensure that lengths of surface geodesics between the vertices of any given triangle do not differ by more than a factor of three, as shown in Figure 9.

The independence property of the disk covering ensures that the minimum surface geodesic length between two interacting vertices is greater or equal to the smaller of the radii of the disks centered at

those vertices. This is true because a vertex has to lay outside of the disk neighborhood of all the other vertices with which it interacts. The constraint on the difference between the radii of the disk neighborhoods of two interacting vertices limits the maximum length of the geodesic between them, which is bound by the sum of the two radii, to three times the length of smaller radius.

Limiting the aspect ratios of the surface geodesics between the vertices of a triangle does not directly induce a similar restriction on the aspect ratio of the triangle itself. The relationship between the two is governed by the integral of the surface curvature over that triangular domain. We chose not to employ any curvature constraints in the first stage. The reason for doing this is that it allows us to generate much coarser meshes, that can serve as domains for parameterization of the subsequently more refined meshes that use curvature constraints. However, by constraining the aspect ratios of the surface geodesics this way we build a coarse base mesh that when refined will converge to triangles whose aspect ratios have magnitude between 1/3 and 1.

3.8 Algorithm Robustness

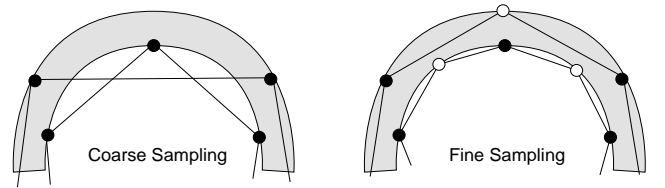


Figure 10: **Algorithm Convergence:** Low resolution meshes may self-intersect. The self-intersections disappear at higher resolutions. Methods based on deformable models cannot automatically “fix” self-intersections.

When extracting thin-shelled iso-surfaces, the low resolution meshes generated may self-intersect, as shown in Figure 10. These self-intersections automatically disappear in the subsequent higher resolution meshes. Most algorithms using deformable models cannot overcome this problem and generate incorrect results. Although it is possible to detect self-intersection at run time we have not implemented it because it would slow down the computations considerably.

3.9 Discussion

It is well known that there is no lower bound on the distance between two nodes of a voxel. As a consequence there is no lower bound on the distance between neighboring candidate vertices. A c-vertex can have neighbors that are very close, as well as neighbors that are much farther away. This can cause problems and generate incomplete covers or triangles with bad aspect ratio if the size of the disk neighborhoods is allowed to be arbitrarily small. Imagine a c-vertex v whose neighbors are all distance 1 away, except for one who is only 0.1 away. If we allow the disk neighborhood at v to have radius 0.2 for example, then the neighbor that is 0.1 away will be inside this disk neighborhood and cannot be a disk center. The closest possible disk centers are at the c-vertices 1 away from v . The constraints on disk radii force us to place disks of radius at most 0.4 here. This generates an incomplete cover because the newly added disk will not intersect the disk of radius 0.2 at v , since the sum of their radii is less than the distance between their centers: $0.2 + 0.6 < 1$. If the radius of the disk at v is allowed to further shrink below 0.1, the triangles produced by the Voronoi algorithm will have edges of length 1 and 0.1, with an aspect ratio of 0.1, which is below our desired lower bound of 0.3.

This problem creates an interesting trade-off between the ability to bound triangle aspect ratios and to always resolve the correct

topology. The lower bound on the triangle aspect ratio can be easily enforced by requiring that all neighboring c-vertices of a disk center be inside that disk. This restriction also ensures that the covering is complete and that there are no holes. However, the surface may exhibit very small details that cannot be resolved when this restriction is enforced. It is possible that a disk will produce a topological event and yet it cannot shrink any further because neighboring vertices would then lie outside the disk. Such a case is present in the brain dataset, and is seen in the inset of Figure 1. The thin protrusion shown cannot be resolved with nice aspect ratio triangles from the volume at its original resolution, and a supersampling factor of 8 is required to preserve acceptable aspect ratios, see Figure 13. Fortunately, this kind of situations can be detected at runtime and can be used to signal the need for local supersampling of the volume. Supersampling would add extra vertices in the neighborhood of the disk center and thus allow it to assume smaller radii while preserving the aspect ratio properties.

4 Results

We present several examples of meshes generated using our algorithm. The examples illustrate the capabilities of our algorithm: automatic coarse adaptive mesh extraction with correct topology and guaranteed minimum triangle aspect ratio, followed by several adaptively refined meshes using the error metrics discussed above. The final high resolution results required approximately 3 seconds for the head model with no super-sampling and 10 minutes for the dragon dataset super-sampled 4 times to an effective resolution of $1424 \times 644 \times 1004$. All computations were performed on an Intel Pentium III 933MHz computer with 512MB of RAM.

Simple Data Set: Scan-Converted Model of a Human Head.

The original volume (resolution: $134 \times 160 \times 186$) in this example was generated by a scan-conversion method that converts polygonal models into a distance volume described in [15]. The coarse mesh extraction took 1 second and generated 38 triangles, Figure 11 left. Note that even this very coarse mesh preserves some of the character of the head model. It is possible for someone familiar with the model to recognize the orientation of the head from this coarse mesh. The two intermediate meshes contain 358 and 4,286 triangles, Figure 11 middle, respectively. The first refinement was done using the maximum distance metric. The second refinement used the curvature metric. Note how the two metrics act differently. The maximum distance metric produces more uniform meshes, while the curvature based metric tends to refine more in areas where the surface folds. The final refinement to 8,752 triangles, Figure 11 right, required an additional 11.1 seconds on a P3-933 machine and used a combination of the two metrics. If the volume is super-sampled two times to an effective resolution of $268 \times 320 \times 372$, the corresponding execution times increase to 4.3 seconds for the coarse mesh extraction and an additional 29.6 seconds for the final refined mesh. The Marching Cubes mesh extracted from the volume at its original resolution of $134 \times 160 \times 186$ contains 107,796 triangles and required 2.5 seconds. Histograms illustrating the distribution of triangle aspect ratios for our final mesh versus the Marching Cubes mesh can be found in Figure 12.

Complex Data Set 1: MRI Scan of a Human Brain. The original volume (resolution: $88 \times 121 \times 62$) in this example was computed from a diffusion tensor MRI scan of a human brain [27]. The iso-surface encloses regions of high diffusion anisotropy, namely the white matter and the corpus callosum. Typical of most medical scans, the surface has high genus, a wide distribution of curvature, and many disconnected components of different sizes. The multiresolution meshes presented in Figure 13, contain 10,606, 16,354, 37,002, and 49,734 triangles respectively. The volume has been super-sampled eight times prior to mesh extraction to ensure correct aspect ratios and topological genus at the same time. Coarse

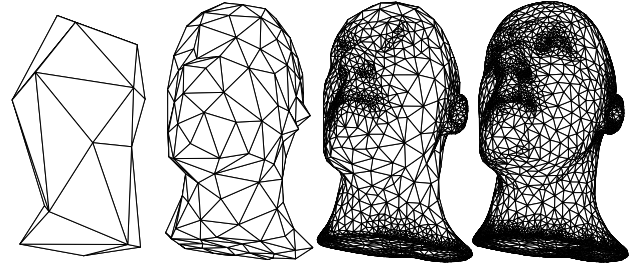


Figure 11: Scan-Converted Model of a Human Head: left to right, 38, 358, 4,286, and 8,752 triangles.

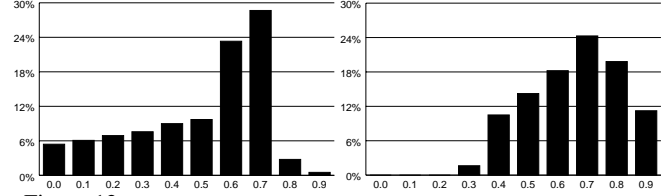


Figure 12: Triangle Aspect Ratio Histograms of extracted meshes of the head model using: left - Marching Cubes, right - our method. Note the nice distribution of the triangle aspect ratios in the meshes generated using our method, and the lack of triangles with aspect ratio below 1/3.

mesh extraction from the volume with no supersampling required 3.6 seconds. When super-sampled 4 times to an effective resolution of $352 \times 484 \times 248$, coarse mesh extraction requires 61.1 seconds, while extracting the final refined mesh takes an additional 89.6 CPU seconds to compute. The Marching Cubes mesh extracted from the volume at its original resolution contains 81,392 triangles.

Complex Data Set 2: Electron Tomogram of a Dendrite.

The original volume (resolution: $154 \times 586 \times 270$) in this example is a level set segmentation [24] of an electron tomogram of a dendrite. The multiresolution meshes presented in Figure 14 contain 3,974, 12,898, 21,648 and 63,906 triangles respectively. The volume has been super-sampled four times to an effective resolution of $616 \times 2342 \times 1080$. Note how all the branches of the dendrite are automatically preserved even in the coarsest mesh and how the resolution adapts to fit the features of the surface. The Marching Cubes mesh extracted from the volume at its original resolution contains 555,912 triangles.

Complex Data Set 3: Laser Range Scan of a Dragon Figurine.

The original volume (resolution: $356 \times 161 \times 251$) in this example was produced by a volumetric laser scan merging algorithm [3] applied to numerous laser range scans of a dragon figurine. The multiresolution meshes presented in Figure 15 contain 4,020, 18,498, 41,910 and 65,420 triangles respectively. The volume has been super-sampled four times, to an effective resolution of $1424 \times 644 \times 1004$. Extraction of similar meshes from volumes super-sampled 2 times ($712 \times 322 \times 502$) took 63 seconds for the coarsest mesh and an additional 163 seconds for the finest resolution mesh. The areas of high concentration of small triangles is due to the presence of small artifacts in the data, the result of imperfect alignment of the different views during the merging stage of the scanning process. The Marching Cubes mesh extracted from the volume at its original resolution contains 438,260 triangles.

5 Future Work

Future work includes several items:

- The algorithm can be extended to generate tetrahedral volumetric meshes with similar guarantees: adaptivity, nice aspect ratios, correct topology. We are not aware of algorithms that can extract such meshes at the present time.

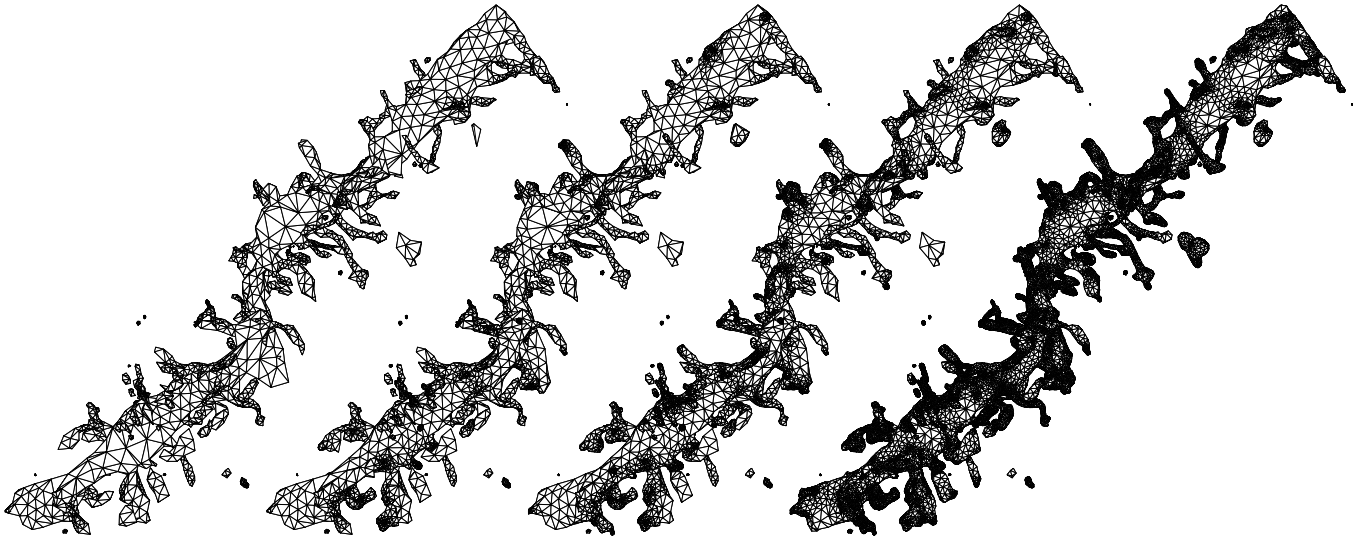


Figure 14: **Electron Tomogram of a Dendrite:** from left to right, coarse mesh - 3,974 triangles, intermediate meshes - 12,898 and 21,648 triangles, final mesh - 63,906 triangles.

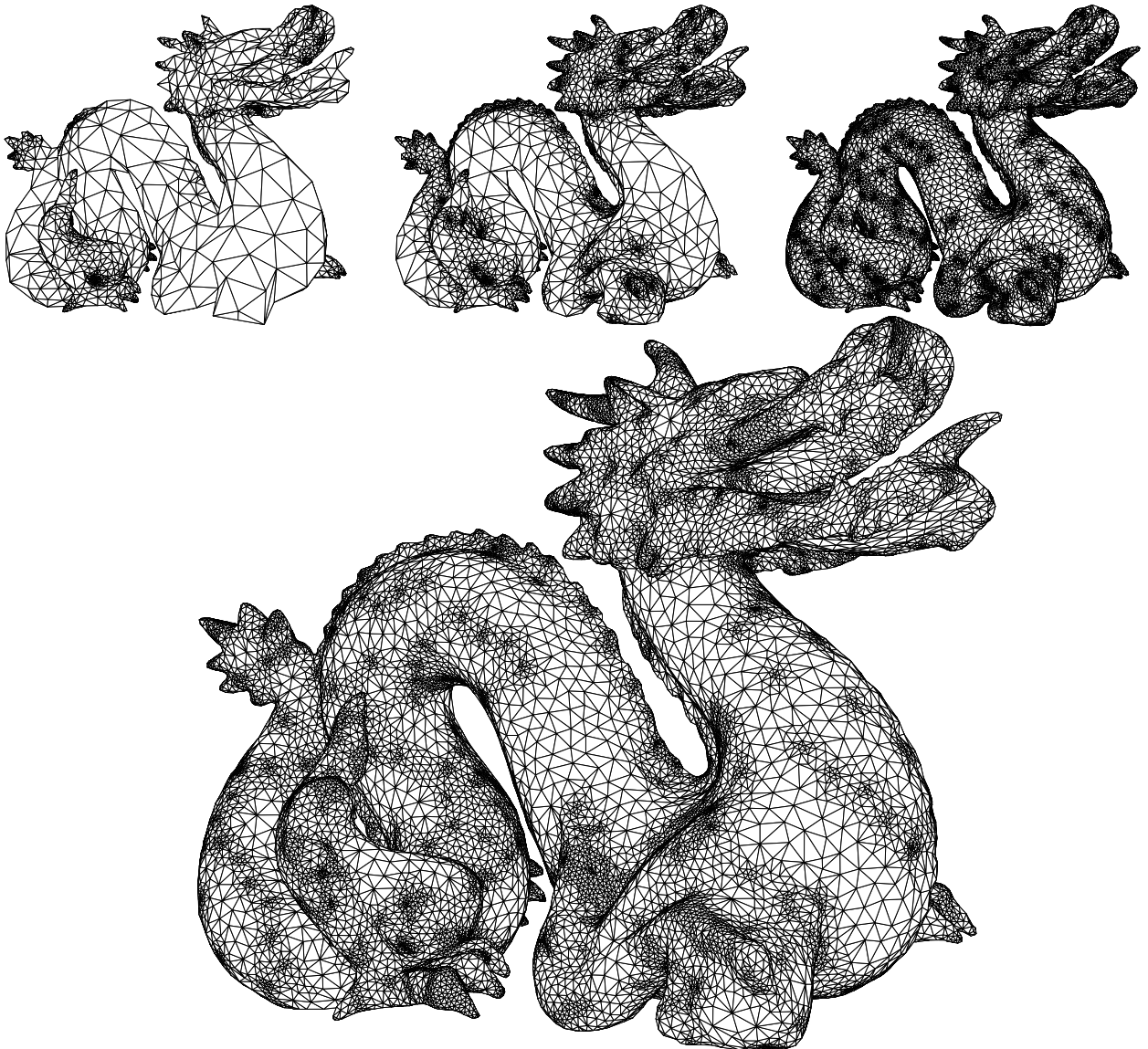


Figure 15: **Laser Range Scan of a Dragon Figurine:** left to right, coarse mesh - 4,020 triangles, intermediate meshes - 8,498 and 41,910 triangles, final mesh - 65,420 triangles.

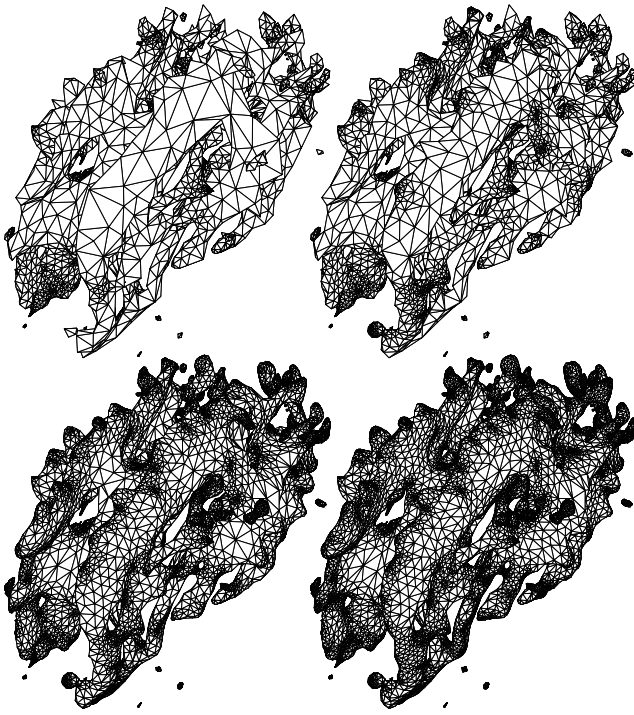


Figure 13: **Diffusion Tensor MRI Scan of a Human Brain:** from left to right, coarse mesh - 10,606 triangles, intermediate meshes - 16,354 and 37,002 triangles, final mesh - 49,734 triangles.

- Supersampling of the volume dataset produces meshes with better aspect ratio triangles in the areas of high resolution. However, uniform supersampling is unnecessary and slows down the algorithm. An algorithm for doing adaptive on-the-fly supersampling of the volume seems feasible and would solve the problems introduced by uniform supersampling.
- Modifications in the algorithm for building the disk covering and for refinement may be able to generate meshes with subdivision connectivity.

6 Acknowledgements

We would like to thank Mathieu Desbrun, Mark Ellisman, Gordon Kindlmann, Maryann Martone, Sean Mauch, Ken Museth, Cici Koenig, Ross Whitaker, Leonid Zhukov, and the Stanford University Computer Graphics Lab for their assistance and support. This work was supported by National Science Foundation grants #ASC-89-20219 and #ACI-9982273; the Office of the Director of Defense Research and Engineering, and the Air Force Office of Scientific Research (F49620-96-1-0471), as part of the MURI program; and the Caltech SURF Program.

References

- [1] J. Bloomenthal. An Implicit Surface Polygonizer. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, July 1994.
- [2] A. Bottino, W. Nuij, and K. van Overveld. How to Shrinkwrap Through a Critical Point: An Algorithm for the Adaptive Tesselation of Iso-surfaces with Arbitrary Topology. In *Implicit Surfaces '96*, pages 53–72, 1996.
- [3] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH '96 Conference Proceedings*, pages 303–312, August 1996.
- [4] Manfredo P. DoCarmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Inc.
- [5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *SIGGRAPH 95 Conference Proceedings*, pages 173–182, July 1995.
- [6] M. Garland and P. Heckbert. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH '97 Conference Proceedings*, pages 209–216, August 1997.
- [7] E. Hartmann. A Marching Method for the Triangulation of Surfaces. *The Visual Computer*, 14(3):95–108, 1998.
- [8] M. Henderson. Computing Implicitly Defined Surfaces: Two Parameter Continuation. Technical Report RC-18777, IBM T. J. Watson Research Center, March 1993.
- [9] H. Hoppe. Progressive Meshes. In *SIGGRAPH '96 Conference Proceedings*, pages 99–108, August 1996.
- [10] T. Karkanis and A.J. Stewart. High Quality, Curvature Dependent Triangulation of Implicit Surfaces. *IEEE Computer Graphics and Applications*, 21(2):60–69, March 2001.
- [11] L. P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum*, 18(3):119–130, July 1998.
- [12] J.-O. Lachaud and A. Montanvert. Deformable Meshes with Automated Topology Changes for Coarse-to-fine 3D Surface Extraction. *Medical Image Analysis*, 3(2):107–207, 1999.
- [13] A.W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *SIGGRAPH '98 Conference Proceedings*, pages 95–104, July 1998.
- [14] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proceedings SIGGRAPH '87)*, 21(4):163–169, July 1987.
- [15] S. Mauch. A Fast Algorithm for Computing the Closest Point and Distance Transform. Submitted to SIAM Journal on Scientific Computing.
- [16] T. McInerney and D. Terzopoulos. Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850, 1999.
- [17] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M. J. Wozny. Geometrically Deformable Models: A Method for Extracting Closed Geometric Models from Volume Data. *Computer Graphics (Proceedings SIGGRAPH '91)*, 25(4):217–226, July 1991.
- [18] H. Müller and M. Stark. Adaptive Generation of Surfaces in Volume Data. *The Visual Computer*, 9(4):182–199, 1993.
- [19] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of Triangle Meshes. *Computer Graphics (Proceedings SIGGRAPH '92)*, 26(2):65–70, July 1992.
- [20] G. Turk. Re-tiling Polygonal Surfaces. In *SIGGRAPH '94 Conference Proceedings*, pages 55–64, 1994.
- [21] K. van Overveld and B. Wyvill. Potentials, Polygons and Penguins. An Efficient Adaptive Algorithm For Triangulating an Equi-potential Surface. In *Proc. 5th Annual Western Computer Graphics Symposium (SKIGRAPH 93)*, pages 31–62, 1993.
- [22] L. Velho, L.H. de Figueiredo, and J. Gomes. A Unified Approach for Hierarchical Adaptive Tesselation of Surfaces. *ACM Transactions on Graphics*, 18(4):329–360, October 1999.
- [23] R. Westermann, L. Kobbelt, and T. Ertl. Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Isosurfaces. *The Visual Computer*, 15(2):100–111, 1999.
- [24] R. Whitaker, D. Breen, K. Museth, and N. Soni. A Framework for Level Set Segmentation of Volume Datasets. In *Proceedings of the International Workshop on Volume Graphics*, pages 159–168, June 2001.
- [25] Z.J. Wood. Semi-Regular Mesh Extraction from Volumes. In *Proceedings of Visualization 2000*, pages 275–282, 2000.
- [26] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structures for Soft Objects. *The Visual Computer*, 2:227–234, 1986.
- [27] L. Zhukov, K. Museth, D. Breen, and R. Whitaker. 3D Modeling and Segmentation of Diffusion Weighted MRI Data. In *Proceedings of SPIE Medical Imaging Conference*, pages 401–412, February 2001.